

A Failure-Distance Based Method to Bound the Reliability of Non-Repairable Fault-Tolerant Systems without the Knowledge of Minimal Cuts

Víctor Suñé and Juan A. Carrasco, *Member, IEEE*

Abstract—CTMC (continuous-time Markov chains) are a commonly used formalism for modeling fault-tolerant systems. One of the major drawbacks of CTMC is the well-known state-space explosion problem. This paper develops and analyzes a method (SC-BM) to compute bounds for the reliability of nonrepairable fault-tolerant systems in which only a portion of the state space of the CTMC is generated. SC-BM uses the failure distance concept as the method described in [1] but, unlike that method, which is based on the computation of exact failure distances, SC-BM uses lower bounds for failure distances, which are computed on the system fault-tree, avoiding the computation and holding of all minimal cuts as required in [1]. This is important because computation of all minimal cuts is NP-hard and the number of minimal cuts can be very large. In some cases SC-BM gives exactly the same bounds as the method in [1]; in other cases it gives less tight bounds. SC-BM computes tight bounds for the reliability of quite complex systems with an affordable number of generated states for short to quite large mission times. The analysis of several examples seems to show that the bounds obtained by SC-BM appreciably outperform those obtained by simpler methods, e.g., [2], and, when they are not equal, are only slightly worse than the bounds obtained by the method in [1]. In addition, the overhead in CPU time due to computing lower bounds for failure distances seems to be reasonable.

Index Terms—Bounds, fault-tolerant system, nonrepairable system, state-space reduction.

ACRONYMS¹

CTMC	continuous-time Markov chain
DTMC	discrete-time Markov chain
LB	lower bound
UB	upper bound
BM-1	bounding method in [1]
SC-BM	bounding method in this paper
T-SC-BM	implementation of T-BM in this paper
FIFO	first in, first out

Definitions:

- bag: collection of possibly repeated elements. The notation $c_1[n_1]c_2[n_2] \cdots c_k[n_k]$ is for a bag a including $n_i > 0$ instances of element c_i , $i = 1, 2, \dots, k$; each $c_i[n_i]$ is a

part of a . Notation for bags is from [4] except that: 1) a subbag b of bag a is $b \subset a$, whether b is strictly contained in a or not, and 2) bags are denoted as explained here

- minimal cut: minimal bag of component classes whose failure implies the system failure
- failure bag: bag of component classes which can fail simultaneously (in a single transition)
- failure distance from state a : minimum number of components which must fail, in addition to those already failed in a , to fail the system

Notation:

$\text{ur}(t)$	unreliability: $\Pr\{\text{system has failed by time } t\}$
$[\text{ur}(t)]_{\text{lb}}$	LB for $\text{ur}(t)$ obtained with SC-BM, BM-1 or T-SC-BM
$[\text{ur}(t)]_{\text{ub}}$	UB for $\text{ur}(t)$ obtained with SC-BM
$[\text{ur}(t)]'_{\text{ub}}$	UB for $\text{ur}(t)$ obtained with BM-1
$[\text{ur}(t)]''_{\text{ub}}$	UB for $\text{ur}(t)$ obtained with T-SC-BM
$\text{poim}(r; \mu)$	$(\mu^r / r!) \cdot \exp(-\mu)$
X	$\{X(t); t \geq 0\}$: acyclic CTMC modeling the system
X'	$\{X'(t); t \geq 0\}$: acyclic CTMC used in SC-BM for computing $[\text{ur}(t)]_{\text{lb}}$ and $[\text{ur}(t)]_{\text{ub}}$
X''	$\{X''(t); t \geq 0\}$: acyclic CTMC used in T-SC-BM for computing $[\text{ur}(t)]_{\text{lb}}$ and $[\text{ur}(t)]''_{\text{ub}}$
λ_a	output rate of state a in X
$\lambda_{a,b}$	transition rate from state a to state b in X
$\lambda_{a,B}$	$\sum_{b \in B} \lambda_{a,b}$
Λ	randomization rate (\geq the maximum output rate of an state of both X and X')
Y	$\{Y_n; n = 0, 1, \dots\}$: DTMC obtained by [3] randomizing X with rate Λ
Y'	$\{Y'_n; n = 0, 1, \dots\}$: DTMC obtained by [3] randomizing X' with rate Λ
O	up states: set of states of X in which the system is operational
f	down state: absorbing state that represents system failure
G	subset of O that is generated
o	state of O without failed components
U	$O - G$
$d(a)$	failure distance from state a
L	$d(o)$
U_d	$\{a \in U d(a) = d\}$
U^k	$\{a \in U \text{number of failed components in } a \text{ is } k\}$
$U_{k,d}$	$U^k \cap U_d$
$\tilde{d}(a)$	LB for $d(a)$
\tilde{L}	$\tilde{d}(o)$

Manuscript received July 9, 1998; revised December 8, 1999. This work was partially supported by the "Comisión Interministerial de Ciencia y Tecnología" (CICYT) under the research Grant TIC 95-0707-C 02-02.

The authors are with the Dep't d'Enginyeria Electrònica, UPC, Diagonal 647 plta. 9, 08028 Barcelona, Spain (e-mail: sunye@eel.upc.es; carrasco@eel.upc.es).

Publisher Item Identifier S 0018-9529(01)06805-1.

¹The singular and plural of an acronym are always spelled the same.

$\tilde{U}_{d,i}$	$\{a \in U_d \tilde{d}(a) = i\}$
\tilde{U}_d	$\{a \in U \tilde{d}(a) = d\}$
$F(a)$	bag of failed component classes in state a
MC	set of minimal cuts of the system
$ \cdot $	cardinality of a set or a bag [4]
E	set of a failure bags of the system
E_i	$\{e \in E e = i\}$
FC	set of different cardinalities of failure bags
$\lambda_{ub}(e)$	UB for the rate with which failure bag $e \in E$ is realized from any state
f_i	$\sum_{e \in E_i} \lambda_{ub}(e)$
\wedge, \vee	logical operator AND, OR.

I. INTRODUCTION

MODELING is important in the design and analysis of fault-tolerant systems. These systems exhibit a stochastic behavior and, therefore, probabilistic measures are adequate for their quantitative assessment. An important class of such systems are those whose components cannot be repaired. For these systems the reliability, $\Pr\{\text{system has not failed by time } t\}$, or its complement, $\text{ur}(t)$, are suitable measures. Non-repairable fault-tolerant systems can be modeled using combinatorial methods and, more generally, hierarchical methods [5], [6]. Hierarchical methods require the behavior of components and subsystems to be mutually s-independent. Recently, combinatorial methods have been improved, allowing some complex dependencies such as lack of coverage [7], [8] to be dealt with. However, when the failure rate of a component depends on the global state of the system, then state-level modeling techniques such as CTMC are required. A major drawback of CTMC, especially of those for modeling complex systems, is that the size of their state space is typically so large that it goes far beyond the available computing resources. This well-known problem is referred to as state-space explosion. One approach to attack this problem is the use of bounding methods, in which only a subset G of O is generated. Typically, G includes the states with up to a given number K of failed components. Bounds for $\text{ur}(t)$ can be trivially derived (e.g., [2]) by modifying X so that exits of X from G not going to f are directed to an absorbing state u_0 . The probability of the modified CTMC X'' being in state f at time t is a LB for $\text{ur}(t)$, and the probability of X'' being in $\{f, u_0\}$ at time t is an UB for $\text{ur}(t)$. This LB is usually good, but the UB is not, because it is equivalent to assuming that the system is nonoperational in all the states in U , which can be far from reality. A more recent paper [1] proposed the BM-1 method, in which the behavior of the system out of the generated portion is bounded using the failure distance concept, resulting in an improved UB for $\text{ur}(t)$.

BM-1 requires computing the failure distances from the states of U that are reachable from G in a single transition. These computations can be done knowing MC. We have developed an algorithm [9] which computes MC efficiently in many cases. However, computing MC is NP-hard [10] and in some cases the algorithm [9] can break down. In addition, $|MC|$ can be very large, causing a large memory overhead due to the need to hold MC

and related data structures for efficient failure distances computation.

This paper develops and analyzes SC-BM, our new bounding method for $\text{ur}(t)$ using LB for failure distances which are obtained on the fault tree, avoiding the computation and holding of MC.

Section II describes the class of models assumed in SC-BM and shows how bounds for $\text{ur}(t)$ are computed in SC-BM from LB for failure distances satisfying some conditions.

Section III defines the LB for failure distances used in SC-BM, proves that they satisfy the required conditions, describes the procedures used in SC-BM to compute such LB, and describes how the CTMC X' is generated in SC-BM.

Section IV describes how the CTMC X'' is generated in T-SC-BM and proves that the cost (in terms of CPU time) of SC-BM is at most identical to the cost of T-SC-BM when $\tilde{L} = 1$.

Section V analyzes SC-BM using two examples, and compares it with T-SC-BM and BM-1.

II. CLASS OF MODELS AND UNRELIABILITY BOUNDS

We consider acyclic CTMC X modeling nonrepairable fault-tolerant systems. We assume that the system is made up of components which can be grouped into classes, the components of the same class being indistinguishable from a dependability view-point. The operational/down state of the system is determined by the unfailed/failed state of its components by a fault tree. The fault tree of the system is constructed using AND and OR gates and inputs. Inputs have associated with them different bags of the form $c[n]$. Input x with associated bag $c[n]$ has the value 1 iff at least n components of class c are failed. The value of the fault tree is computed as usual from the values of its inputs. The system is down iff the value of the fault tree is 1. To avoid trivialities, we assume that no inputs x, y with associated bags $c[n], c[n']$, $n \neq n'$ feed the same gate. This is not a true restriction because, for $n' > n$ and an OR gate, x, y can be replaced by x and for an AND gate by y . Each state $a \in O$ has associated with it a bag of failed component classes $F(a)$. There is a single state, state o , with $F(o) = \emptyset$. Each transition of X has associated with it a failure bag $e \in E$, including the components which are failed when the transition is followed. Imperfect coverage can be modeled by introducing fictitious components which do not fail by themselves and to which uncovered faults are propagated. This point is illustrated by the following example.

Fig. 1 shows the architecture of an example system, adapted from [6], which is used for illustration. The system consists of

- 2 memory modules MM_1 and MM_2 ,
- 3 CPU chips CPUC,
- 2 port chips PTC.

In addition, to model imperfect coverage, 1 fictitious component RMM_j , $j = 1, 2$, and 2 fictitious components RCM are added to the system. One CPUC and 1 PTC are spares. Each MM_j has 10 memory chips MC_j , 2 of which are spares, and 1 interface chip IC_j . The IC_j and active MC_j , PTC and CPUC fail, respectively, with rate $\lambda_{IC_j}, \lambda_{MC_j}, \lambda_{PTC}$ and λ_{CPUC} . Spare chips fail with rates $\nu \cdot \lambda_{MC_j}, \nu \cdot \lambda_{PTC}$ and $\nu \cdot \lambda_{CPUC}$, where $\nu, 0 < \nu < 1$

TABLE I
FAILURE BAGS OF THE EXAMPLE SYSTEM AND, FOR EACH FAILURE BAG e , A SUITABLE UB $\lambda_{ub}(e)$

	Description	$\lambda_{ub}(e)$
e_1	$MC_1[1]$	$(8 + 2\nu) \cdot \lambda_{MC_1} \cdot C_{MC}$
e_2	$MC_1[1]$ $RMM_1[1]$	$(8 + 2\nu) \cdot \lambda_{MC_1} \cdot (1 - C_{MC}) \cdot C_{MM}$
e_3	$MC_1[1]$ $RMM_1[1]$ $RCM[2]$	$(8 + 2\nu) \cdot \lambda_{MC_1} \cdot (1 - C_{MC}) \cdot (1 - C_{MM})$
e_4	$IC_1[1]$	$\lambda_{IC_1} \cdot C_{MM}$
e_5	$IC_1[1]$ $RCM[2]$	$\lambda_{IC_1} \cdot (1 - C_{MM})$
e_6	$MC_2[1]$	$(8 + 2\nu) \cdot \lambda_{MC_2} \cdot C_{MC}$
e_7	$MC_2[1]$ $RMM_2[1]$	$(8 + 2\nu) \cdot \lambda_{MC_2} \cdot (1 - C_{MC}) \cdot C_{MM}$
e_8	$MC_2[1]$ $RMM_2[1]$ $RCM[2]$	$(8 + 2\nu) \cdot \lambda_{MC_2} \cdot (1 - C_{MC}) \cdot (1 - C_{MM})$
e_9	$IC_2[1]$	$\lambda_{IC_2} \cdot C_{MM}$
e_{10}	$IC_2[1]$ $RCM[2]$	$\lambda_{IC_2} \cdot (1 - C_{MM})$
e_{11}	$CPUC[1]$	$(2 + \nu) \cdot \lambda_{CPUC} \cdot C_{CPUC}$
e_{12}	$CPUC[1]$ $RCM[2]$	$(2 + \nu) \cdot \lambda_{CPUC} \cdot (1 - C_{CPUC})$
e_{13}	$PTC[1]$	$(1 + \nu) \cdot \lambda_{PTC} \cdot C_{PTC}$
e_{14}	$PTC[1]$ $RCM[2]$	$(1 + \nu) \cdot \lambda_{PTC} \cdot (1 - C_{PTC})$

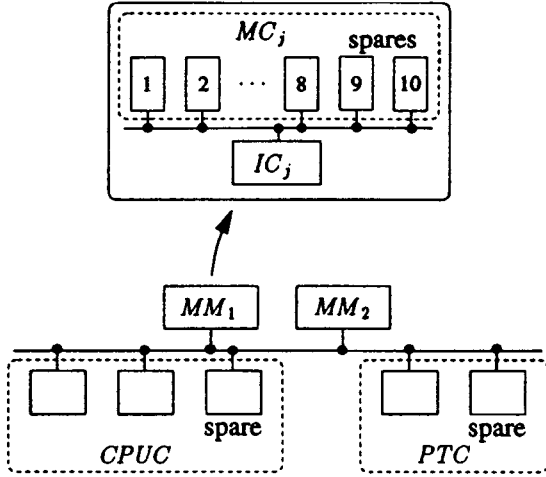


Fig. 1. Architecture of the example system.

is a dormancy factor. Recovery is hierarchical. A fault in a MC_j is covered with probability C_{MC} . Failure of MM_j and faults of $CPUC$ and PTC are covered with probabilities C_{MM} , C_{CPUC} and C_{PTC} , respectively. To model imperfect coverage, an uncovered fault in a MC_j is propagated to the fictitious component RMM_j , and an uncovered failure of MM_j , and an uncovered fault of a $CPUC$ or a PTC are propagated to the 2 fictitious components RCM . The MM_j is operational if at least 8 MC_j , the IC_j and the RMM_j are unfailed. The system is operational if at least 1 memory module is operational, and at least 2 $CPUC$, 1 PTC and 1 RCM are unfailed.

Table I gives the failure bags of the example system and, for each failure bag e , a suitable $\lambda_{ub}(e)$ expressed in terms of the above failure rates and coverage probabilities, and the dormancy factor ν . Thus, e.g.,

- e_1 is the fault of a memory chip of the first memory module which is covered at memory module level,
- e_2 is the fault of that chip which is uncovered at memory module level and covered at system level,
- e_3 is the uncovered fault of the chip.

For the example system,

$$FC = \{1, 2, 3, 4\}$$

and

$$f_1 = \lambda_{ub}(e_1) + \lambda_{ub}(e_4) + \lambda_{ub}(e_6) + \lambda_{ub}(e_9) + \lambda_{ub}(e_{11}) + \lambda_{ub}(e_{13}),$$

$$f_2 = \lambda_{ub}(e_2) + \lambda_{ub}(e_7),$$

$$f_3 = \lambda_{ub}(e_5) + \lambda_{ub}(e_{10}) + \lambda_{ub}(e_{12}) + \lambda_{ub}(e_{14}),$$

$$f_4 = \lambda_{ub}(e_3) + \lambda_{ub}(e_8).$$

A. SC-BM

SC-BM computes $[\text{ur}(t)]_{lb}$ and $[\text{ur}(t)]_{ub}$ by solving the transient regime of the CTMC X' . The CTMC X' has state space $G \cup \{f\} \cup \{u_0, \dots, u_{\tilde{L}}\}$. Although other selections for G are possible, we assume that G includes all the up states of X with up to K failed components. We also assume $\Pr\{X(0) \in G\} = 1$. The states u_d , $0 \leq d \leq \tilde{L}$ pessimistically approximate the behavior of X from the instant in which X enters U from G . The transition rates in X'

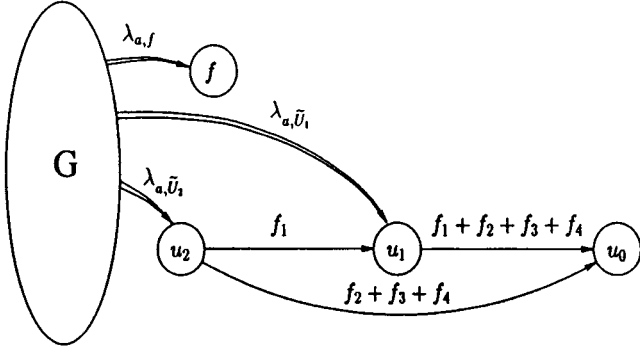
- from a to b , with $a, b \in G$, and
- from a to f , with $a \in G$

are as in X . The transition rates from states $a \in G$ to u_d , with $1 \leq d \leq \tilde{L}$ have values λ_a, \tilde{u}_d , and for each $1 \leq d \leq \tilde{L}$ and each $i \in FC$, there is a transition rate f_i from u_d to $u_{\max\{0, d-i\}}$. The initial probability distribution of X' in G is the same as the initial probability distribution of X in G . Section III-A shows that $\tilde{L} = 2$ for the example system. Fig. 2 shows the structure of X' for the example system. The bounds are:

$$[\text{ur}(t)]_{lb} = \Pr\{X'(t) = f\},$$

$$[\text{ur}(t)]_{ub} = \Pr\{X'(t) \in \{u_0, f\}\}. \quad (1)$$

The correctness of $[\text{ur}(t)]_{lb}$ is trivial. The correctness of $[\text{ur}(t)]_{ub}$ is proved in Section II-B. Given the relationships


 Fig. 2. State transition diagram of X' for the example system.

between X' and the CTMC used in BM-1 [1], it is easy to conclude that SC-BM and BM-1 give exactly the same bounds when $\tilde{d}(a) = d(a)$, $a \in U$. Otherwise, BM-1 gives, in general, tighter bounds than SC-BM.

B. Correctness of $[\text{ur}(t)]_{\text{ub}}$

This section establishes the correctness of $[\text{ur}(t)]_{\text{ub}}$ under the conditions $1 \leq \tilde{d}(a) \leq d(a)$, $a \in U$ and $\tilde{d}(a) \leq \tilde{L}$.

The proof is constructed with the aid of the DTMC Y and Y' . Since [11] $X = \{X(t); t \geq 0\}$ is probabilistically identical to $\{Y_{N(t)}; t \geq 0\}$ and $X' = \{X'(t); t \geq 0\}$ is probabilistically identical to $\{Y'_{N(t)}; t \geq 0\}$, where $N = \{N(t); t \geq 0\}$ is a Poisson process with arrival rate Λ independent of both Y and Y' :

$$\Pr\{X(t) = a\} = \sum_{n=0}^{\infty} \text{poin}(n; \Lambda \cdot t) \cdot \Pr\{Y_n = a\}, \quad (2)$$

$$\Pr\{X'(t) = a\} = \sum_{n=0}^{\infty} \text{poin}(n; \Lambda \cdot t) \cdot \Pr\{Y'_n = a\}. \quad (3)$$

Let

$$R'_m(d) \equiv \Pr\{Y'_m = u_0 | Y'_0 = u_d\}, \quad (4)$$

$$R_m(a) \equiv \Pr\{Y_m = f | Y_0 = a\}. \quad (5)$$

Then we have the following two results.

Lemma 1: $R'_m(d)$, $m > 0$, $d > 0$ is decreasing on d .

Proposition 1: $R_m(a) \leq R'_m(\min\{d, \tilde{L}\})$, $a \in U_d$, $m > 0$, $d > 0$.

Using lemma 1 and proposition 1, it is possible to prove proposition 2, which establishes that Y' “upper bounds” Y .

Proposition 2: Let $1 \leq \tilde{d}(a) \leq d(a)$, $a \in U$, $\tilde{d}(a) \leq \tilde{L}$, $\Pr\{X(0) \in G\} = 1$. Then, $\Pr\{Y_n = f\} \leq \Pr\{Y'_n \in \{u_0, f\}\}$, $n > 0$.

Using proposition 2, the correctness of $[\text{ur}(t)]_{\text{ub}}$ can be proved:

Theorem 1: Let $1 \leq \tilde{d}(a) \leq d(a)$, $a \in U$, $\tilde{d}(a) \leq \tilde{L}$, and $\Pr\{X(0) \in G\} = 1$. Then, $\text{ur}(t) \leq [\text{ur}(t)]_{\text{ub}}$.

Lemma 1, which is formally identical to [1, Lemma 1], propositions 1 and 2, and theorem 1 are proved in the Appendix.

III. COMPUTATION OF LB FOR FAILURE DISTANCES AND MODEL GENERATION IN SC-BM

Definitions:

- node (also referred to as event): gate or input of the fault tree
- related: two inputs x, y are related if $b(x) = c[n]$ and $b(y) = c[n']$, $n \neq n'$
- realized: an event x is realized if $\text{val}(x) = 1$
- path: a sequence of nodes $x_1 \dots x_k$ such that $x_i \in \text{fo}(x_{i+1})$, $i = 1, \dots, k-1$
- reachable node: a node x is reachable from node y if there exists a path from y to x
- module: $x \in I \cup P$ is a module iff every path $z \dots y$, $z \notin \text{Reach}(x)$, $y \in \text{Reach}(x)$ contains node x , and for each input $y \in \text{Support}(x)$, no related inputs exist outside $\text{Support}(x)$

Notation:

C	set of component classes
I	set of inputs (basic events) of the fault tree; each input x has associated with it a different bag, $b(x)$, of the form $c[n]$, $c \in C$, $n \geq 1$
P	set of gates (complex events) of the fault tree
g_r	root gate (top event) of the fault tree
$\text{type}(g)$	type of gate g : AND or OR
$\text{fo}(x)$	fanout of (set of nodes fed by) node x
$\text{fi}(x)$	fanin of (set of nodes that feed) node x
$\text{val}(\cdot)$	value (1 or 0) of an input or a gate
$\text{Reach}(x)$	node x plus set of nodes reachable from x
$\text{Support}(x)$	$I \cap \text{Reach}(x)$, $x \in I \cup P$
$S(F, x)$	$ F \cap (\cup_{y \in \text{Support}(x)} b(y)) $, where F is a bag of component classes and x is a node
$d_b(F, x)$	distance from F to x : minimum number of components which must fail in addition to those in the bag of component classes F to realize event x
$\tilde{d}_b(F, x)$	LB for $d_b(F, x)$
$\tilde{\eta}(e)$	$\tilde{d}_b(e, g_r)$: LB for the failure distance from a failure bag $e \in E$ to g_r

This Section III:

- Obtains the LB $\tilde{d}(a) = \tilde{d}_b(F(a), g_r)$ for the failure distances from states a used in SC-BM. The bounds are proved to fulfill the conditions which, according to theorem 1, guarantee the correctness of $[\text{ur}(t)]_{\text{ub}}$ and the condition $\tilde{d}(a) = 0$ iff $d(a) = 0$, which eases the generation of X' (see Section III-D).
- Gives sufficient conditions for $\tilde{d}(a) = d(a)$.
- Describes procedures which can be used to compute the LB for the failure distances from the successors of a state, and describes how the CTMC X' is generated in SC-BM using those procedures.

A. Recursive Definition of LB for Failure Distances

The LB for failure distances $\tilde{d}(a) = \tilde{d}_b(F(a), g_r)$ used in SC-BM are defined on the fault tree of the system using the concept of module, which generalizes to component classes the definition in [12], [13], in the sense that a module is a node such that the subtree hanging from it has that node as only entry point and every input of the subtree does not have related inputs

TABLE II
COMPUTATION OF $\tilde{L} = \tilde{d}_b(\emptyset, g_r)$ TRAVERSING DEPTH-FIRST LEFT-MOST THE FAULT TREE OF THE EXAMPLE SYSTEM STARTING AT g_r AND USING (6)

step	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
node x	g_r	g_1	g_2	x_1	x_2	x_3	g_2	g_3	x_4	x_5	x_6	g_3	g_1	x_7	x_8	x_9	g_r
$\tilde{d}_b(\emptyset, x)$	—	—	—	3	1	1	1	—	3	1	1	1	2	2	2	2	2

outside the subtree. To determine which gates or inputs of the fault tree are modules, the algorithm LTA/DR [13] is used with a small modification to deal with component classes: during the first depth-first left-most traversal of the fault tree (step 2 of the algorithm LTA/DR), visit to $x \in I$ implies simultaneous visit (*viz.*, with the same time-stamp as for x) to all inputs related to it.

Given a bag of component classes F , $\tilde{d}_b(F, x)$, $x \in I \cup P$ is defined recursively by:

For $x \in I$, $b(x) = c[n]$:

$$\tilde{d}_b(F, x) \equiv \begin{cases} n & \text{if no } c[n'] \text{ is part of } F \\ \max\{0, n - n'\} & \text{if } c[n'] \text{ is part of } F. \end{cases} \quad (6)$$

For $x \in P$, $\text{type}(x) = \text{OR}$:

$$\tilde{d}_b(F, x) \equiv \min_{y \in \text{fi}(x)} \{\tilde{d}_b(F, y)\}. \quad (7)$$

For $x \in P$, $\text{type}(x) = \text{AND}$:

$$\begin{aligned} \tilde{d}_b(F, x) \equiv & \sum_{y \in A(x)} \tilde{d}_b(F, y) \\ & + \max \left\{ \sum_{y \in B(x)} \tilde{d}_b(F, y), \right. \\ & \left. \max_{y \in C(x)} \{0, \tilde{d}_b(F, y)\} \right\}, \end{aligned} \quad (8)$$

$$A(x) \equiv \{y \in \text{fi}(x) | y \text{ is a module} \wedge |\text{fo}(y)| = 1\},$$

$$B(x) \equiv \{y \in \text{fi}(x) | y \text{ is a module} \wedge |\text{fo}(y)| > 1 \vee y \text{ is not a module} \wedge y \in I\},$$

$$C(x) \equiv \{y \in \text{fi}(x) | y \text{ is not a module} \wedge y \in P\}.$$

Eqs. (6)–(8) allow computation of $\tilde{d}_b(F, g_r)$ traversing the fault tree depth-first left-most, starting at g_r . Fig. 3 shows the fault tree of the example system. Table II shows how $\tilde{L} = \tilde{d}_b(\emptyset, g_r)$ is computed for that fault tree. All gates and inputs of the fault tree are modules and, therefore, (8) reduces to:

$$\tilde{d}_b(F, x) = \sum_{y \in \text{fi}(x)} \tilde{d}_b(F, y).$$

B. Correctness of the LB for Failure Distances and Related Results

Theorems 2–4 are proved in [14].

Theorem 2: Let F be a bag of component classes and $x \in I \cup P$. Then, the $\tilde{d}_b(F, x)$, defined recursively by (6)–(8), verify:

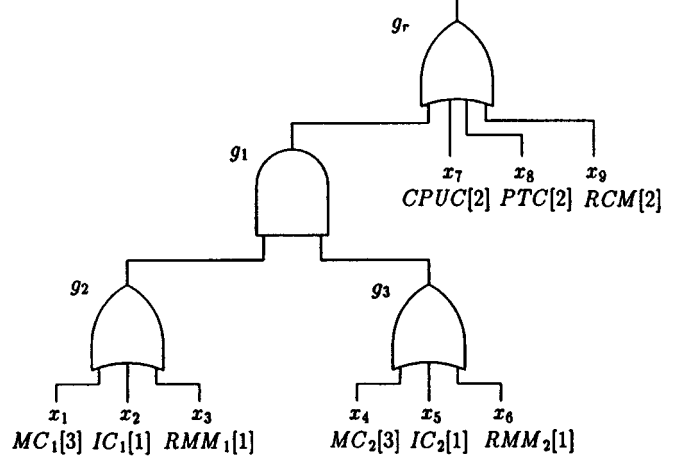


Fig. 3. Fault tree of the example system. The bag associated with an input is given next to that input.

- $0 \leq \tilde{d}_b(F, x) \leq d_b(F, x)$.
- $\tilde{d}_b(F, x) = 0$ iff $d_b(F, x) = 0$.

Theorem 3: Let $x \in I \cup P$ and let F be a bag of component classes. Then, $\tilde{d}_b(F, x) = d_b(F, x)$ if for every $z \in P$ with $\text{type}(z) = \text{AND}$ one of the following two conditions holds:

- Every $y \in \text{fi}(z)$ is a module or an input.
- There exists only one $u \in \text{fi}(z)$ which is neither a module nor an input and every $y \in \text{fi}(z)$, $y \neq u$ is a module with $|\text{fo}(y)| = 1$.

Theorem 4: Let F, F', F'' be bags of component classes with $F = F' + F''$ and let $x \in I \cup P$. Then,

$$\tilde{d}_b(F', x) \geq \tilde{d}_b(F, x) \geq \tilde{d}_b(F'', x) - S(F'', x).$$

Let a be a state. With $F = F(a)$ and $x = g_r$, part b of theorem 2 implies $\tilde{d}(a) = 0$ iff $d(a) = 0$, part a implies $0 \leq \tilde{d}(a) \leq d(a)$, and both results imply $1 \leq \tilde{d}(a) \leq d(a)$ for $a \in U$. In addition, taking $x = g_r$, $F' = \emptyset$, $F'' = F(a)$, and $F = F' + F'' = F(a)$, the left inequality of theorem 4 states that $\tilde{L} = \tilde{d}_b(\emptyset, g_r) \geq \tilde{d}_b(F(a), g_r) = \tilde{d}(a)$. Thus, the derived $\tilde{d}(a)$ satisfy the requirements of theorem 1, guaranteeing that the upper bound $[\text{ur}(t)]_{\text{ub}}$ obtained by SC-BM is correct. In addition, we have $\tilde{d}(a) = 0$ iff $d(a) = 0$. With $F = F(a)$ and $x = g_r$, theorem 3 gives a sufficient condition for $\tilde{d}(a) = d(a)$ which can be checked inexpensively. Finally, when $L = 1$, theorem 2 with $F = F(a)$ and $x = g_r$ implies $\tilde{d}(a) = d(a)$. Thus, when the condition of theorem 3 is satisfied or $L = 1$, $\tilde{d}(a) = d(a)$ and SC-BM gives the same bounds as BM-1.

C. Computation of the LB for Failure Distances

Section III-D shows that the generation of the CTMC X' can be done knowing $\tilde{d}(b)$ for all successors b of each state $a \in G$. Each $\tilde{d}(b)$ could be computed from $F(b)$ by traversing the

fault tree as described in Section III-A. However, that procedure is expensive if the fault tree is large. This section describes a typically much more efficient procedure, `comp_all_d`, which can compute $\tilde{d}(b)$ for all successors b of a state.

The procedure `comp_all_d` is built on top of three procedures: `update_d`, `comp_d`, `restore_d`.

Each node x of the fault tree holds a distance variable $dv(x)$ properly initialized. The procedure `update_d` takes as inputs a bag of component classes F , a positive integer ub and a stack CS , and processes the fault tree as follows. For each $c[n]$ that is part of F , the procedure makes $dv(x) = \max\{0, dv(x) - n\}$ for each input x , $b(x) = c[n]$ and follows a recursive processing from x . The recursive processing from a node z of the fault tree involves computing for each $y \in fo(z)$ a potential new value for $dv(y)$ using (7) for OR gates and (8) for AND gates with $dv(t)$, $t \in fi(y)$ instead of $\tilde{d}_b(F, t)$. If the potential new value for $dv(y)$ is $< ub$ and smaller than the current value of $dv(y)$, the variable $dv(y)$ is updated and the processing of the fault tree continues recursively from node y . When a distance variable is updated, the corresponding node and the old value of the variable are put in CS .

The procedure `comp_d` takes as inputs a bag of component classes F , two nonnegative integers lb and ub , with $lb \leq ub$, and a stack CS . If $lb = ub$, the procedure returns lb without doing anything else. If $lb < ub$, the procedure makes the call `update_d(F, ub, CS)` and returns $\min\{dv(g_r), ub\}$.

The procedure `restore_d` takes as input a stack CS and simply restores the distance variable of the nodes kept in CS to their old values. After the call to `restore_d`, CS becomes empty.

The procedure `comp_all_d` takes as inputs a bag of failed component classes F , the LB $\tilde{d} = \tilde{d}_b(F, g_r)$, a subset E' of the set of failure bags E , and $\tilde{\eta}(e)$, $e \in E'$, and computes $\tilde{d}_b(F + e, g_r)$ for each $e \in E'$. For the procedure to work properly, \tilde{d} must be > 0 and the distance variable $dv(x)$ of each node x of the fault tree must have been initialized to $\tilde{d}_b(\emptyset, x)$. The procedure is:

```

Let CS and CS' be empty stacks.
comp_d(F, 0,  $\tilde{d}$ , CS);
for (each  $e \in E'$ ) {
   $s = \max\{0, \tilde{d} - |e|, \tilde{\eta}(e) - |F|\}$ ;
   $t = \min\{\tilde{d}, \tilde{\eta}(e)\}$ ;
   $\tilde{d}_b(F + e, g_r) = \text{comp\_d}(e, s, t, CS')$ ;
  restore_d(CS');
}
restore_d(CS);

```

After calling the procedure, the distance variable $dv(x)$ of each node x of the fault tree has the value $\tilde{d}_b(\emptyset, x)$. The following two results are from [13]. Theorem 5 is used in Section III-D to justify the algorithm for the generation of X' in SC-BM. Theorem 6 asserts the correctness of `comp_all_d`.

Theorem 5: Let F be a bag of component classes, let lb and ub be nonnegative integers with $lb \leq \tilde{d}_b(F, g_r) \leq ub$, and let CS be a stack. Let the distance variable $dv(x)$ of each

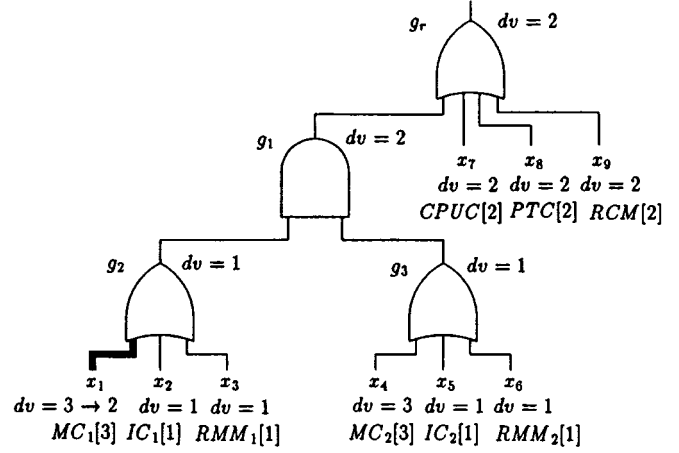


Fig. 4. Processing of the fault tree of the example system during the call `update_d(F = MC1[1], ub = 2, CS)`. The distance variable values are given next to each node. The “ \rightarrow ” shows their changes.

node x of the fault tree be initialized to $\tilde{d}_b(\emptyset, x)$. Then, the call `comp_d(F, lb, ub, CS)` returns $\tilde{d}_b(F, g_r)$.

Theorem 6: Let F be a bag of component classes, let $\tilde{d} = \tilde{d}_b(F, g_r) > 0$, and let $E' \subset E$. Let the distance variable $dv(x)$ of each node x of the fault tree be initialized to $\tilde{d}_b(\emptyset, x)$. Then, the call `comp_all_d(F, \tilde{d} , E' , $\tilde{\eta}(e)$, $e \in E'$)` computes correctly the lower bounds $\tilde{d}_b(F + e, g_r)$, $e \in E'$.

This Section III-C ends by illustrating the procedure `comp_all_d` using the example system of Section II, whose fault tree is in Fig. 3. The procedure is illustrated for the inputs $F = MC_1[1]$, $\tilde{d}_b(F, g_r) = 2$ and $E' = \{e_4, e_7\}$ with $e_4 = IC_1[1]$, $e_7 = MC_2[1]RMM_2[1]$, $\tilde{\eta}(e_4) = 1$, and $\tilde{\eta}(e_7) = 1$ (see Table I). The procedure begins by creating empty stacks CS and CS' and making the call `comp_d(F = MC1[1], lb = 0, ub = 2, CS)`. Since $lb < ub$, that call to `comp_d` results in the call `update_d(F = MC1[1], ub = 2, CS)`. Fig. 4 illustrates the processing of the fault tree during that call. The $dv(x)$ of each node x is initialized to $\tilde{d}_b(\emptyset, x)$ before the call. The only input having associated with it a bag related to $MC_1[1]$ is x_1 . Using (6), $dv(x_1)$ is updated to $\max\{0, 3 - 1\} = 2$ and the pair $(x_1, 3)$ is stored in CS . The change in $dv(x_1)$ is not propagated up the fault tree because the new value of $dv(x_1)$ is not smaller than $ub = 2$. Next, the procedure processes the failure bag $e_4 = IC_1[1]$. The values of s, t are $s = 1$ and $t = 1$; the procedure continues by making the call `comp_d(F = IC1[1], lb = 1, ub = 1, CS')` followed by the call `restore_d(CS')`. The first call returns $lb = 1$ without processing the fault tree because $lb = ub$. This results in $\tilde{d}_b(MC_1[1]IC_1[1], g_r) = 1$. The second call does not restore any distance variable because $CS' = \emptyset$. The procedure continues by processing the failure bag $e_7 = MC_2[1]RMM_2[1]$. The values of s, t become $s = 0$ and $t = 1$ and the procedure continues by making the call `comp_d(F = MC2[1]RMM2[1], lb = 0, ub = 1, CS')`. Since $lb < ub$, the call to `comp_d` results in the call `update_d(F = MC2[1]RMM2[1], ub = 1, CS')`. Fig. 5 illustrates the processing of the fault tree during that call. The only inputs having associated with them a bag related to some part of $MC_2[1]RMM_2[1]$ are x_4, x_6 . Using (6), $dv(x_4)$ is

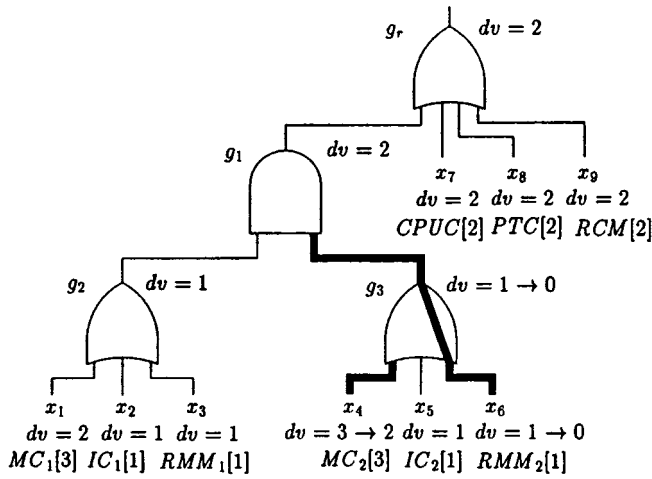


Fig. 5. Processing of the fault tree of the example system during the call `update_d(F = MC2[1]RMM2[1], ub = 1, CS)`. The distance variable values are given next to each node. The “ \rightarrow ” shows their changes.

updated to 2 and the pair $(x_4, 3)$ is stored in CS' . The change in $dv(x_4)$ is not propagated up the fault tree because the new value of $dv(x_4)$ is not smaller than $ub = 1$. Next, the input x_6 is dealt with. Using (6), $dv(x_6)$ is updated to 0 and the pair $(x_6, 1)$ is stored in CS' . Since the new value of $dv(x_6)$ is smaller than $ub = 1$, that change is propagated up the fault tree to gate g_3 . Using (7), the potential new value for $dv(g_3)$ is 0. Since that potential new value is smaller than both the old value of the variable and $ub = 1$, $dv(g_3)$ is updated, the pair $(g_3, 1)$ is stored in CS' and the change is propagated up the fault tree to gate g_1 . Using (8), the potential new value for $dv(g_1)$ is 1, which is not smaller than $ub = 1$. Then, $dv(g_1)$ is not updated, and because there is no pending processing of the fault tree, the call `update_d(F = MC2[1]RMM2[1], ub = 1, CS')` ends and the call `comp_d(F = MC2[1]RMM2[1], lb = 0, ub = 1, CS')` returns the value $\min\{dv(g_r), ub\} = \min\{2, 1\} = 1$. This results in $\tilde{d}_b(MC_1[1]MC_2[1]RMM_2[1], g_r) = 1$. The procedure finishes by making the calls `restore_d(CS')` and `restore_d(CS)`, which leave the fault tree with the $dv(x)$ of each node x equal to $\tilde{d}_b(\emptyset, x)$.

D. Generation of the CTMC X' in SC-BM

To describe with detail the generation of the CTMC X' in SC-BM, let the failure behavior of the fault-tolerant system be described by a high-level specification made up of a set of rules. Each rule is composed of a guard condition that determines when the rule is enabled in a given (current) state, an execution part that allows to obtain a next state from the current one, and a rate specification that determines the rate with which the next state is reached. Each rule r has associated with it a failure bag $e(r)$ meaning that the rule models the failure of the components in $e(r)$. Different rules can have associated with them the same failure bag. We assume the availability of three procedures named `enabled`, `rate` and `successor`, which provide the interface with the high-level specification required for model generation. The procedure `enabled` has as inputs a state a and returns the set of rules $R(a)$ enabled in a . The procedure `rate` takes as inputs a state a and a rule r and returns the rate

$\lambda^{r,a}$ with which the next state is reached from a following rule r . The procedure `successor` takes as inputs a state a and a rule r and returns the state b reached from a following r . Although the interface with the high-level specification could be provided by other sets of procedures, the assumed one is reasonable, matches our implementation, and leads to an efficient generation of X' , allowing a fair comparison between SC-BM and T-SC-BM.

The \tilde{L} and $\tilde{\eta}(e)$, $e \in E$ are computed before the generation of X' . To do that, first the $dv(x)$ of each node x of the fault tree is initialized to $\tilde{d}_b(\emptyset, x)$ by traversing the fault tree depth-first left-most, starting at g_r and using (6)–(8). Since $\tilde{L} = \tilde{d}_b(\emptyset, g_r)$, after the initialization $dv(g_r)$ holds \tilde{L} . The $\tilde{\eta}(e)$, $e \in E$ are computed using `comp_d` and `restore_d`. Note that $\tilde{\eta}(e) = \tilde{d}_b(e, g_r) \geq 0$ and, according to the definition of $\tilde{d}_b(\cdot)$, $\tilde{\eta}(e) \leq \tilde{d}_b(\emptyset, g_r) = \tilde{L}$. This allows the computation of $\tilde{\eta}(e)$, $e \in E$ by creating an empty stack CS and, for each $e \in E$, making the call `comp_d(e, 0, \tilde{L}, CS)` followed by the call `restore_d(CS)`. Using theorem 5 with $F = e$, $lb = 0$ and $ub = \tilde{L}$, the value returned by each call `comp_d(e, 0, \tilde{L}, CS)` is $\tilde{\eta}(e)$. After computing $\tilde{\eta}(e)$, $e \in E$, the distance variable of each node x of the fault tree holds its initial value $\tilde{d}_b(\emptyset, x)$.

The CTMC X' is generated breadth-first using \tilde{L} , $\tilde{\eta}(e)$, $e \in E$, and a FIFO queue Q , by calling `comp_all_d`, `enabled`, `rate`, and `successor`. The queue Q holds triplets $(s, F(s), \tilde{d}(s))$ corresponding to the states s that have to be processed. The generation of X' begins by creating the states f and u_i , $0 \leq i \leq \tilde{L}$, adding the transition rates f_i from u_d to $u_{\max\{0, d-i\}}$ (see Section II-A), and making $Q = \{(o, \emptyset, \tilde{L})\}$ and $G = \{o\}$. Then, while $Q \neq \emptyset$, a triplet $(a, F(a), \tilde{d}(a))$ is pulled out of Q and processed as follows. First, $R(a)$ is obtained by making the call `enabled(a)`. Next, letting $E(a) = \{e(r)\}_{r \in R(a)}$, the LB $\tilde{d}(a, e) = \tilde{d}_b(F(a) + e, g_r)$ for the failure distance from the states with bag of failed component classes $F(a) + e$, $e \in E(a)$ are obtained by making the call `comp_all_d(F(a), \tilde{d}(a), E(a), \tilde{\eta}(e), e \in E(a))`. The LB for the failure distance from the state reached from a following rule $r \in R(a)$ is $\tilde{d}(a, e(r))$. Then, each $r \in R(a)$ is processed as follows. 1) $\lambda^{r,a}$ is computed by making the call `rate(a, r)`. 2) If $\tilde{d}(a, e(r)) = 0$, the state reached from a following r is a down state and a transition rate $\lambda^{r,a}$ from a to f is added. If $\tilde{d}(a, e(r)) > 0$ and $|F(a) + e(r)| > K$, the state reached from a following r belongs to \tilde{U}_d , $d = \tilde{d}(a, e(r))$ and a transition rate $\lambda^{r,a}$ from a to u_d is added. 3) If $\tilde{d}(a, e(r)) > 0$ and $|F(a) + e(r)| \leq K$, the state reached from a following r belongs to the subset of states that have to be generated. In that case, the reached state, b , is obtained by making the call `successor(a, r)`; if $b \notin G$, b is added to G , a transition rate $\lambda^{r,a}$ from a to b is added and the triple $(b, F(a) + e(r), \tilde{d}(a, e(r)))$ is put into Q ; if $b \in G$, a transition rate $\lambda^{r,a}$ from a to b is added.

IV. GENERATION OF X'' IN T-SC-BM AND COMPARISON OF SC-BM WITH T-SC-BM

This section describes T-SC-BM, and shows that, when $\tilde{L} = 1$, the cost (in terms of CPU time) of SC-BM is at most identical to the cost of T-SC-BM.

T-BM requires knowledge of the operational/down state of the successors b of an operational state a . T-SC-BM uses `eval_all_ft` to determine, using the fault tree, whether the states reached from a given state following the rules enabled in it are operational or down. The procedure `eval_all_ft` is built on top of two procedures: `eval_ft` and `restore_ft`.

Each input x of the fault tree has associated with it a distance variable $dv(x)$, and each node z of the fault tree has associated with it a value variable $vv(z)$. The procedure `eval_ft` takes as inputs a bag of failed component classes F and a stack CS , and works as follows. For each input x , $b(x) = c[n]$ for which $c[n']$ is part of F , $dv(x)$ is set to $\max\{0, dv(x) - n'\}$ and, if $dv(x)$ becomes 0, x is implied to 1 (i.e., $vv(x)$ is set to 1). Each implication to 1 of an input x is propagated up the fault tree while the visited gate becomes implied to 1. When a distance variable changes, the corresponding input and the old value of the variable are put in CS . Gates that become implied to 1 are put in CS too. The procedure returns $vv(g_r)$.

The procedure `restore_ft` has as input a stack CS . For each input x of the fault tree held in CS , the procedure restores $dv(x)$ to its old value, setting $vv(x)$ to 0 if the restored value is > 0 . For each gate x kept in CS , the procedure sets $vv(x)$ to 0. After the call to `restore_ft`, CS becomes empty.

The procedure `eval_all_ft` takes as inputs a bag of failed component classes F and a subset $E' \subset E$, and for each $e \in E'$, computes the value, v_{F+e} , of the root gate of the fault tree when the bag of failed component classes is $F + e$. For the procedure to work properly, the $dv(x)$ of each input x , $b(x) = c[n]$ of the fault tree must have been initialized to n , and the $vv(x)$ of each node of the fault tree must have been set to 0. The procedure is as follows.

```

Let CS and CS' be empty stacks.
eval_ft(F, CS);
for (each  $e \in E'$ ) {
     $v_{F+e} = \text{eval\_ft}(e, CS')$ ;
    restore_ft(CS');
}
restore_ft(CS);
    
```

After calling the procedure, $dv(x) = n$ for each input x , $b(x) = c[n]$ of the fault tree and $vv(z) = 0$ for each node z of the fault tree.

The CTMC X'' is generated breadth-first using a FIFO queue Q and calling the procedure `eval_all_ft` and the procedures enabled, `rate`, `successor` described in Section III-D. The queue Q holds pairs $(s, F(s))$ corresponding to the states s that have to be processed. The generation of X'' begins by creating the states f and u_0 , and making $Q = \{(o, \emptyset)\}$ and $G = \{o\}$. Then, while $Q \neq \emptyset$, a pair $(a, F(a))$ is pulled out of Q and processed as follows. 1) The set of rules enabled in a , $R(a)$, is obtained by making the call `enabled(a)`. 2) Denoting by $e(r)$ the failure bag associated with rule r and letting $E(a) = \{e(r)\}_{r \in R(a)}$, the values of the root gate of the fault tree, $v_{F(a)+e}$, for the states with bag of failed component classes $F(a) + e$, $e \in E(a)$ are obtained by making the call `eval_all_ft(F(a), E(a))`. The value of the root gate of the fault tree for the state reached from a following rule $r \in R(a)$

is $v_{F(a)+e(r)}$. 3) Each $r \in R(a)$ is processed as follows. 3a) The rate $\lambda^{r,a}$ with which the next state is reached from a following rule r is computed by making the call `rate(a, r)`. 3b) If $v_{F(a)+e(r)} = 1$, then the state reached from a following r is a down state and a transition rate $\lambda^{r,a}$ from a to f is added. If $v_{F(a)+e(r)} = 0$ and $|F(a) + e(r)| > K$, the state reached from a following r belongs to U and a transition rate $\lambda^{r,a}$ from a to u_0 is added. 3c) If $v_{F(a)+e(r)} = 0$ and $|F(a) + e(r)| \leq K$, the state reached from a following r belongs to the subset of states that have to be generated. In that case, the reached state, b , is obtained by making the call `successor(a, r)` and, if $b \notin G$, b is added to G , a transition rate $\lambda^{r,a}$ from a to b is added, and the pair $(b, F(a) + e(r))$ is put into Q ; if $b \in G$, a transition rate $\lambda^{r,a}$ from a to b is added.

This section ends by comparing the cost (in terms of CPU time) of SC-BM with the cost of T-SC-BM for the particular case $\tilde{L} = 1$. The comparison is done with the aid of theorem 7, proved in [14].

Theorem 7: Let F be a bag of component classes, let $\tilde{d} = \tilde{d}_b(F, g_r) > 0$, and let $E' \subset E$. If $\tilde{L} = 1$, then the cost of the call `comp_all_d(F, \tilde{d} , E' , $\tilde{\eta}(e)$, $e \in E'$)` after setting $dv(x) = \tilde{d}_b(\emptyset, g_r)$, $x \in I \cup P$ is at most equal to the cost of the call `eval_all_ft(F, E')` after setting $dv(x) = n$, $x \in I$, $b(x) = c[n]$, and $val(x) = 0$, $x \in I \cup P$.

The subset G generated by both methods is the same. Then, taking into account the description of the generation of X' in SC-BM done in Section III-D and the description of the generation of X'' in T-SC-BM done in this section, the former differs basically from the latter only in that a call to `comp_all_d` is done during the generation of X' whenever a call to `eval_all_ft` is done during the generation of X'' . Then, since solution of X' and X'' takes negligible time compared with their generation, comparison of the costs of both methods can be done approximately by comparing, for a given $a \in G$ and $E(a) \subset E$, the cost of the call `comp_all_d(F(a), $\tilde{d}(a)$, $E(a)$, $\tilde{\eta}(e)$, $e \in E(a)$)` done in SC-BM with the cost of the corresponding call `eval_all_ft(F(a), $E(a)$)` done in T-SC-BM. Therefore, invoking theorem 7 with $F = F(a)$, $\tilde{d} = \tilde{d}_b(F(a), g_r) = \tilde{d}(a) > 0$ and $E' = E(a)$, it follows that, for the particular case $\tilde{L} = 1$, the cost of SC-BM is at most equal to the cost of T-SC-BM.

V. ANALYSIS AND COMPARISON

This section analyzes SC-BM and compares it with BM-1 and T-SC-BM by means of two large examples representing two scenarios:

- 1) The fault tree satisfies the condition of theorem 3 and $\tilde{L} > 1$.
- 2) The fault tree does not satisfy the condition of theorem 3 and $\tilde{L} > 1$.

In both examples, the state o is the initial state of X . The CTMC are solved in all methods using the randomization method [15]. In example 1, SC-BM and BM-1 give exactly the same bounds, since theorem 3 guarantees $\tilde{d}(a) = d(a)$. In example 2, SC-BM gives less tight bounds than BM-1. Since $\tilde{L} > 1$ for both examples, SC-BM could have in both examples

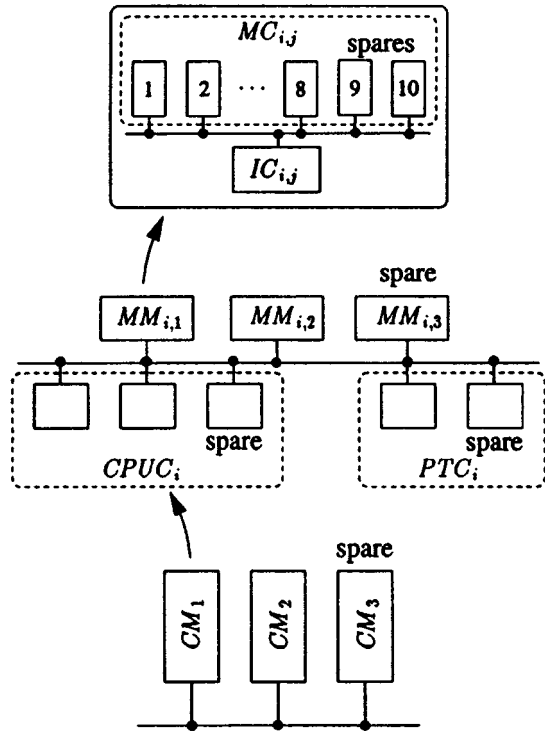


Fig. 6. Architecture of example 1.

a higher cost (in terms of CPU time) than T-SC-BM. The parameter R of the algorithms for computing failure distances used in BM-1 [1] was set to 2 for both examples.

A. Example 1 (Scenario 1)

The system, adapted from [6], has 114 components, and its architecture is shown in Fig. 6. The system has 3 computing modules CM_i , $1 \leq i \leq 3$, one of which is spare. The CM_i module includes:

- 3 memory modules $MM_{i,j}$, $1 \leq j \leq 3$,
- 3 identical CPU chips $CPUC_i$,
- 2 identical port chips PTC_i .

1 memory module, 1 $CPUC_i$ and 1 PTC_i are spares. The $MM_{i,j}$ has:

- 10 identical memory chips $MC_{i,j}$, 2 of which are spares,
- 1 interface chip $IC_{i,j}$.

Memory modules $MM_{i,j}$ and $MM_{i',j}$ are identical (e.g., $MM_{1,1}$ and $MM_{2,1}$).

Also:

- Active $MC_{i,j}$ and active $IC_{i,j}$ fail, respectively, with rates λ_{MC_j} and λ_{IC_j} ;
- Active PTC_i and active $CPUC_i$ fail, respectively, with rates λ_{PTC_i} and λ_{CPUC_i} .
- Spare chips fail with rates $\nu \cdot \lambda_{MC_j}$, $\nu \cdot \lambda_{IC_j}$, $\nu \cdot \lambda_{PTC_i}$, $\nu \cdot \lambda_{CPUC_i}$, where ν , $0 < \nu < 1$ is a dormancy factor.

Components of nonoperational memory modules and nonoperational computing modules do not fail.

Recovery is hierarchical. A fault in a $MC_{i,j}$ is covered with probability C_{MC} . Failure of $MM_{i,j}$, a $CPUC_i$ and a PTC_i is successfully covered with probabilities C_{MM} , C_{CPUC} and C_{PTC} , respectively. Failure of CM_i is covered with probability C_{CM} .

Coverage faults are modeled by introducing fictitious components as explained in Section II:

- An uncovered fault in a $MC_{i,j}$ is propagated to a fictitious component $RMM_{i,j}$.
- An uncovered failure in $MM_{i,j}$, a $CPUC_i$ or a PTC_i is propagated to 2 fictitious components RCM_i .
- An uncovered failure in CM_i is propagated to 4 fictitious components $RSYS$.

Also:

- $MM_{i,j}$ is operational if at least 8 $MC_{i,j}$, the $IC_{i,j}$ and the $RMM_{i,j}$ are unfailed.
- CM_i is operational if at least 2 memory modules are operational and 2 $CPUC_i$, 1 PTC_i , and 1 RCM_i are unfailed.
- The system is operational if at least 2 computing modules are operational and 1 $RSYS$ is unfailed.

The fault tree has:

- 37 inputs, all of which are modules,
- 25 gates, 13 of which are modules,
- 73 edges.

The fault tree is defined by the logical expressions:

$$FM_{i,j} = T_{i,j} \vee U_{i,j} \vee V_{i,j}, \quad 1 \leq i, j \leq 3;$$

$$FMM_{i,l,k} = FM_{i,l} \wedge FM_{i,k}, \quad 1 \leq i \leq 3, \\ 1 \leq l < k \leq 3;$$

$$FC_i = FMM_{i,1,2} \vee FMM_{i,1,3} \vee FMM_{i,2,3} \\ \vee W_i \vee X_i \vee Y_i, \quad 1 \leq i \leq 3;$$

$$FCC_{i,j} = FC_i \wedge FC_j, \quad 1 \leq i < j \leq 3;$$

$$g_r = Z \vee FCC_{1,2} \vee FCC_{1,3} \vee FCC_{2,3}.$$

The bags associated with the inputs of the fault tree are:

$$b(T_{i,j}) = MC_{i,j}[3], \quad b(U_{i,j}) = IC_{i,j}[1], \\ b(V_{i,j}) = RMM_{i,j}[1], \quad b(W_i) = CPUC_i[2], \\ b(X_i) = PTC_i[2], \quad b(Y_i) = RCM_i[2], \\ b(Z) = RSYS[4].$$

Also, $L = \tilde{L} = 4$, $|MC| = 2$, 701.

The numerical results are for the parameters:

$$\lambda_{MC_1} = 10^{-7}h^{-1}, \quad \lambda_{MC_2} = 2 \cdot 10^{-7}h^{-1},$$

$$\lambda_{MC_3} = 3 \cdot 10^{-7}h^{-1}, \quad \lambda_{IC_1} = 2 \cdot 10^{-7}h^{-1},$$

$$\lambda_{IC_2} = 4 \cdot 10^{-7}h^{-1}, \quad \lambda_{IC_3} = 6 \cdot 10^{-7}h^{-1},$$

$$\lambda_{CPUC_1} = 6 \cdot 10^{-7}h^{-1}, \quad \lambda_{CPUC_2} = 1.2 \cdot 10^{-6}h^{-1},$$

$$\lambda_{CPUC_3} = 1.8 \cdot 10^{-6}h^{-1}, \quad \lambda_{PTC_1} = 6 \cdot 10^{-7}h^{-1},$$

$$\lambda_{PTC_2} = 1.2 \cdot 10^{-6}h^{-1}, \quad \lambda_{PTC_3} = 1.8 \cdot 10^{-6}h^{-1},$$

$$\nu = 0.2, \quad C_{MC} = 0.99, \quad C_{MM} = 0.95,$$

$$C_{CPUC} = 0.99, \quad C_{PTC} = 0.97, \quad C_{CM} = 0.95.$$

B. Example 2 (Scenario 2)

The system has 60 components, and its architecture is sketched in Fig. 7. The system has 4 processing clusters that

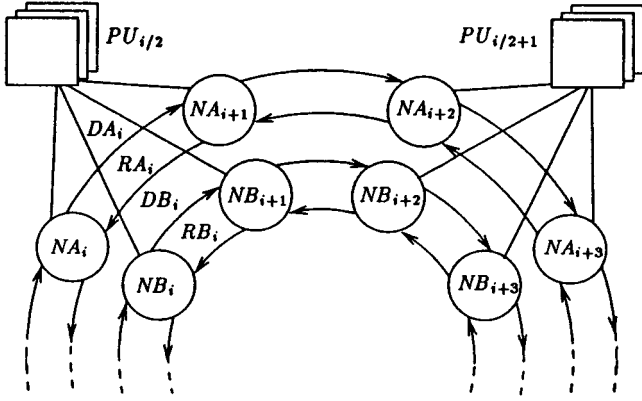


Fig. 7. Architecture of Example 2.

communicate through 2 double-ring networks, A and B; both networks have the same structure.

- Processing cluster i , $0 \leq i \leq 3$ includes 3 processing units PU_i .
- Network A includes 8 nodes NA_i , $0 \leq i \leq 7$. Nodes NA_i and $NA_{(i+1) \bmod 8}$ communicate through direct (clockwise) and reverse (counter-clockwise) links DA_i and RA_i , respectively.
- Network B includes 8 nodes NB_i , $0 \leq i \leq 7$. Nodes NB_i and $NB_{(i+1) \bmod 8}$ communicate through direct and reverse links DB_i and RB_i , respectively.

The operational configuration of the system includes

- 2 processing units from the processing clusters with 2 or 3 unfailed processing units,
- 1 processing unit from the processing clusters with 1 unfailed processing unit,
- the components of network A or B, with priority given to network A, required to build one of the operational network configurations described next. The operational network configurations are, in the order they are tried:
- A direct ring including all nodes and direct links.
- A reverse ring including all nodes and reverse links.
- A configuration, which is used when parallel direct and inverse links i are failed, including all nodes and links except the 2 failed links.
- A configuration, which is used when, for instance, node i fails, including all nodes except node i , and all links except those between node i and nodes $(i \pm 1) \bmod 8$.

When one of these network operational configurations is built, the corresponding network is said to be operational. The components included in the system operational configuration are called active.

- Active processing units, active nodes, and active links fail with rates λ_{PU} , λ_N and λ_L , respectively.
- Inactive processing units fail with rate $\nu \cdot \lambda_{PU}$, where ν , $0 < \nu < 1$, is a dormancy factor.
- Inactive nodes and links of network A fail, respectively, with rate $\nu \cdot \lambda_N$ and $\nu \cdot \lambda_L$ when the network is operational and do not fail otherwise.
- Inactive nodes and links of network B fail, respectively, with rate $\nu \cdot \lambda_N$ and $\nu \cdot \lambda_L$.
- Coverage is perfect for link faults.

- Faults in active processing units and nodes are covered with probabilities C_{PU} and C_N , respectively.
- Coverage faults are modeled by adding 3 fictitious components RSYS as explained in Section II and propagating to all of them any uncovered fault.

The system is operational if the following a, b, c are all true:

- each processing cluster has at least 1 unfailed processing unit,
- one of the previously described operational network configurations for either network A or network B can be built,
- at least 1 RSYS is unfailed.

The system fault-tree has

- 53 inputs, all of which are modules,
- 40 gates, 4 of which are modules,
- 764 edges.

The fault tree is described by:

$$DRA = \bigvee_{i=0}^7 (S_i \vee T_i), \quad DRB = \bigvee_{i=0}^7 (V_i \vee W_i),$$

$$RRA = \bigvee_{i=0}^7 (S_i \vee U_i), \quad RRB = \bigvee_{i=0}^7 (V_i \vee X_i),$$

$$FRA_i = \bigvee_{j=0}^7 S_j \vee \bigvee_{j=0, j \neq i}^7 (T_j \vee U_j),$$

$$FRB_i = \bigvee_{j=0}^7 V_j \vee \bigvee_{j=0, j \neq i}^7 (W_j \vee X_j),$$

$$i^* \equiv (i - 1) \bmod 8,$$

$$FRA_i^* = \bigvee_{j=0, j \neq i}^7 S_j \vee \bigvee_{j=0, j \neq i, i^*}^7 (T_j \vee U_j),$$

$$FRB_i^* = \bigvee_{j=0, j \neq i}^7 V_j \vee \bigvee_{j=0, j \neq i, i^*}^7 (W_j \vee X_j),$$

$$NETA = DRA \wedge RRA \wedge \bigwedge_{i=0}^7 FRA_i \wedge \bigwedge_{i=0}^7 FRA_i^*,$$

$$NETB = DRB \wedge RRB \wedge \bigwedge_{i=0}^7 FRB_i \wedge \bigwedge_{i=0}^7 FRB_i^*,$$

$$NET = NETA \wedge NETB,$$

$$g_r = NET \vee Z \vee \bigvee_{i=0}^3 Y_i.$$

The bags associated with the inputs of the fault tree are:

$$\begin{aligned} b(S_i) &= NA_i[1], & b(T_i) &= DA_i[1], & b(U_i) &= RA_i[1], \\ b(V_i) &= NB_i[1], & b(W_i) &= DB_i[1], & b(X_i) &= RB_i[1], \\ b(Y_i) &= PU_i[3], & b(Z) &= RSYS[3]. \end{aligned}$$

Also, $L = 3$, $\tilde{L} = 2$, $|MC| = 32, 405$.

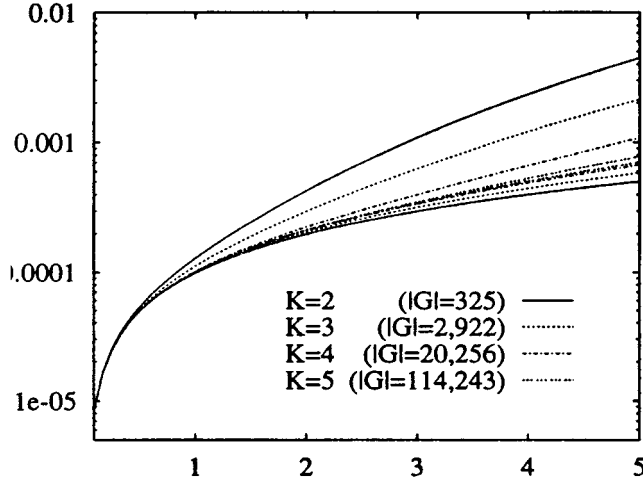


Fig. 8. Unreliability bounds for example 1 as a function of time (years) and K .

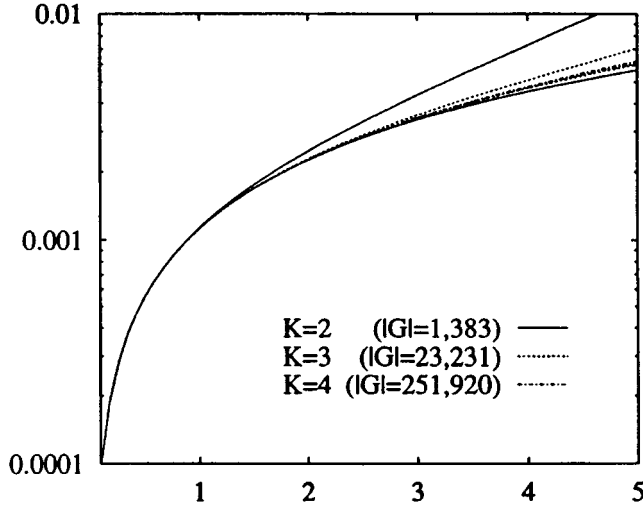


Fig. 9. Unreliability bounds for example 2 as a function of time (years) and K .

The numerical results have been obtained for the parameter values:

$$\lambda_{PU} = 10^{-6} \text{h}^{-1}, \quad \lambda_N = 5 \cdot 10^{-7} \text{h}^{-1}, \quad \lambda_L = 3 \cdot 10^{-7} \text{h}^{-1} \\ \nu = 0.2, \quad C_{PU} = 0.99, \quad C_N = 0.99.$$

C. Results and Discussion

We use $K = 2, 3, 4, 5$ for example 1, and $K = 2, 3, 4$ for example 2.

Figs. 8 and 9 show the unreliability bounds obtained using SC-BM for examples 1 and 2, respectively, as a function of time (in years).² The bounds degrade as time increases. In both examples, however, SC-BM achieves tight bounds for mission times up to 5 years using affordable numbers of states.

²1 month = 730 hours; 1 year = 8760 hours.

TABLE III
EXAMPLE 2: RELATIVE UNRELIABILITY BAND OBTAINED WITH SC-BM, $\text{urb}(t)$ (TOP), AND T-SC-BM, $\text{urb}''(t)$ (BOTTOM)

time	$K (G)$			
	2 (325)	3 (2,922)	4 (20,256)	5 (114,243)
1 month	$5.6074 \cdot 10^{-3}$ $1.9121 \cdot 10^1$	$9.0751 \cdot 10^{-4}$ $5.3599 \cdot 10^{-1}$	$2.4718 \cdot 10^{-5}$ $5.9234 \cdot 10^{-3}$	$2.4558 \cdot 10^{-7}$ $5.1234 \cdot 10^{-5}$
2 months	$1.4916 \cdot 10^{-2}$ $1.9310 \cdot 10^1$	$3.4877 \cdot 10^{-3}$ $7.5534 \cdot 10^{-1}$	$1.1790 \cdot 10^{-4}$ $1.4218 \cdot 10^{-2}$	$1.7226 \cdot 10^{-6}$ $1.9082 \cdot 10^{-4}$
6 months	$8.8675 \cdot 10^{-2}$ $2.0721 \cdot 10^1$	$3.0034 \cdot 10^{-2}$ 1.6262	$1.7085 \cdot 10^{-3}$ $7.0069 \cdot 10^{-2}$	$5.1856 \cdot 10^{-5}$ $2.0870 \cdot 10^{-3}$
1 year	$3.0844 \cdot 10^{-1}$ $2.4590 \cdot 10^1$	$1.1625 \cdot 10^{-1}$ 2.9683	$1.0286 \cdot 10^{-2}$ $2.1744 \cdot 10^{-1}$	$5.2415 \cdot 10^{-4}$ $1.1154 \cdot 10^{-2}$
2 years	1.1610 $3.7307 \cdot 10^1$	$4.4492 \cdot 10^{-1}$ 6.0617	$6.3144 \cdot 10^{-2}$ $7.2137 \cdot 10^{-1}$	$5.5772 \cdot 10^{-3}$ $6.4111 \cdot 10^{-2}$
5 years	7.9242 $9.5814 \cdot 10^1$	2.6360 $1.9943 \cdot 10^1$	$6.0118 \cdot 10^{-1}$ 3.6513	$1.0547 \cdot 10^{-1}$ $6.0539 \cdot 10^{-1}$

TABLE IV
EXAMPLE 2: RELATIVE UNRELIABILITY BAND OBTAINED WITH SC-BM, $\text{urb}(t)$ (TOP), BM-1, $\text{urb}'(t)$ (MIDDLE), AND T-SC-BM, $\text{urb}''(t)$ (BOTTOM)

time	$K (G)$		
	2 (1,383)	3 (23,231)	4 (251,920)
1 month	$8.0995 \cdot 10^{-6}$ $4.2519 \cdot 10^{-6}$ $3.4966 \cdot 10^{-3}$	$2.7765 \cdot 10^{-8}$ $1.9822 \cdot 10^{-8}$ $1.0013 \cdot 10^{-5}$	$5.9186 \cdot 10^{-11}$ $5.1065 \cdot 10^{-11}$ $2.1062 \cdot 10^{-8}$
2 months	$6.4332 \cdot 10^{-5}$ $3.4014 \cdot 10^{-5}$ $1.3816 \cdot 10^{-2}$	$4.3975 \cdot 10^{-7}$ $3.1450 \cdot 10^{-7}$ $7.9310 \cdot 10^{-5}$	$1.8731 \cdot 10^{-9}$ $1.6169 \cdot 10^{-9}$ $3.3356 \cdot 10^{-7}$
6 months	$1.6870 \cdot 10^{-3}$ $9.1670 \cdot 10^{-4}$ $1.1952 \cdot 10^{-1}$	$3.4190 \cdot 10^{-5}$ $2.4625 \cdot 10^{-5}$ $2.0576 \cdot 10^{-3}$	$4.3531 \cdot 10^{-7}$ $3.7652 \cdot 10^{-7}$ $2.5930 \cdot 10^{-5}$
1 year	$1.2903 \cdot 10^{-2}$ $7.2810 \cdot 10^{-3}$ $4.5153 \cdot 10^{-1}$	$5.1391 \cdot 10^{-4}$ $3.7395 \cdot 10^{-4}$ $1.5491 \cdot 10^{-2}$	$1.3015 \cdot 10^{-5}$ $1.1291 \cdot 10^{-5}$ $3.8967 \cdot 10^{-4}$
2 years	$9.4070 \cdot 10^{-2}$ $5.6666 \cdot 10^{-2}$ 1.6114	$7.2234 \cdot 10^{-3}$ $5.3589 \cdot 10^{-3}$ $1.0930 \cdot 10^{-1}$	$3.6184 \cdot 10^{-4}$ $3.1572 \cdot 10^{-4}$ $5.4737 \cdot 10^{-3}$
5 years	1.0991 $7.6243 \cdot 10^{-1}$ 7.1970	$1.8392 \cdot 10^{-1}$ $1.4348 \cdot 10^{-1}$ 1.1300	$2.1999 \cdot 10^{-2}$ $1.9505 \cdot 10^{-2}$ $1.3744 \cdot 10^{-1}$

Tables III and IV compare, for, respectively, examples 1 and 2, and several mission times, the relative unreliability band obtained with SC-BM,

$$\text{urb}(t) = \frac{[\text{ur}(t)]_{\text{ub}} - [\text{ur}(t)]_{\text{lb}}}{[\text{ur}(t)]_{\text{lb}}},$$

against that obtained with BM-1,

$$\text{urb}'(t) = \frac{[\text{ur}(t)]'_{\text{ub}} - [\text{ur}(t)]_{\text{lb}}}{[\text{ur}(t)]_{\text{lb}}},$$

and that obtained with T-SC-BM,

$$\text{urb}''(t) = \frac{[\text{ur}(t)]''_{\text{ub}} - [\text{ur}(t)]_{\text{lb}}}{[\text{ur}(t)]_{\text{lb}}}.$$

The relative band $\text{urb}'(t)$ is not shown in Table III because, for example 1, $[\text{ur}(t)]'_{\text{ub}} = [\text{ur}(t)]_{\text{ub}}$, and, therefore, $\text{urb}'(t) = \text{urb}(t)$.

SC-BM appreciably outperforms T-SC-BM in terms of bounds tightness. Thus, for mission times up to 1 year,

$$\frac{\text{urb}''(t)}{\text{urb}(t)} \geq \begin{cases} 21 & \text{for example 1,} \\ 30 & \text{for example 2.} \end{cases}$$

TABLE V
CPU TIME IN SECONDS TO GENERATE THE CTMC AND COMPUTE THE
UNRELIABILITY BOUNDS FOR $t = 5$ YEARS FOR BOTH EXAMPLES USING
SC-BM (TOP), BM-1 (MIDDLE) AND T-SC-BM (BOTTOM)

Example	K			
	2	3	4	5
#1	0.271	2.12	14.8	90.4
	1.72	4.38	29.3	268
	0.197	1.52	11.0	73.2
#2	3.21	44.8	510	—
	16.4	68.8	998	—
	2.42	36.4	427	—

In addition, SC-BM can compute bounds which are almost as tight or even tighter than those given by T-SC-BM using appreciably fewer states. Thus,

- for example 1 and $t = 2$ years, the relative band obtained by SC-BM with $K = 4$ ($|G| = 20256$) is better than that obtained by T-SC-BM with $K = 5$ ($|G| = 114243$);
- for example 2 and $t = 2$ years, the relative band obtained by SC-BM with $K = 3$ ($|G| = 23231$) is only slightly worse than that obtained by T-SC-BM with $K = 4$ ($|G| = 251920$).

For example 1, SC-BM and BM-1 give exactly the same bounds. For example 2, the bounds obtained by SC-BM are only slightly worse than the bounds obtained by BM-1.

Table V gives the CPU times in seconds for $t = 5$ years for examples 1 and 2 and all 3 methods. The times were measured on a 167 MHz, 128 MB SPARC Ultra 1 workstation. As discussed in Section IV, with respect to T-SC-BM, SC-BM can introduce an appreciable CPU time overhead due to computing LB for failure distances from states only when $\tilde{L} > 1$. That is the case for examples 1 and 2; the results in Table V indicate that the overhead is reasonable, ranging:

- from 20% for example 2 and $K = 4$,
- to 40% for example 1 and $K = 3$.

In addition, SC-BM is always appreciably faster than BM-1.

To conclude, SC-BM seems to give appreciably tighter bounds than T-SC-BM and seems to be only slightly slower than T-SC-BM, when it is not as fast or faster ($\tilde{L} = 1$). Compared with BM-1, when the condition of theorem 3 is satisfied or $L = 1$, then SC-BM is guaranteed to give exactly the same bounds as BM-1, and, in those cases, SC-BM is definitely better, since it does not require the computation of MC, does not incur in the memory overhead associated with the holding of MC and related data structures for failure distance computation [1], and seems to be faster. When the condition of theorem 3 is not satisfied and $L > 1$, SC-BM gives, in general, less tighter bounds than BM-1, but, since SC-BM does not require the computing of MC, which is an NP-hard problem, there are cases in which SC-BM applies while BM-1 does not. In addition, even when MC can be computed, the memory overhead in BM-1 associated with MC is large if $|MC|$ is large, and, then, SC-BM might be better than BM-1 when considering the tradeoff between memory-consumption and bounds-tightness.

APPENDIX

A. Proof of Lemma 1

From the structure of Y' ,

$$R'_m(0) = 1, \quad m > 0, \quad (\text{A-1})$$

$$R'_1(d) = \sum_{i \in \text{FC}, i \geq d} \frac{f_i}{\Lambda}, \quad d > 0. \quad (\text{A-2})$$

Also, for $m > 1, d > 0$,

$$R'_m(d) = \left(1 - \frac{1}{\Lambda} \cdot \sum_{i \in \text{FC}} f_i\right) \cdot R'_{m-1}(d) + \sum_{i \in \text{FC}} \frac{f_i}{\Lambda} \cdot R'_{m-1}(\max\{0, d - i\}). \quad (\text{A-3})$$

The proof is by induction on m .

- Base case ($m = 1$): We show that $R'_1(d) \leq R'_1(d - 1)$, $d > 0$. For $d = 1$, using (A-2), $\Lambda \geq \sum_{i \in \text{FC}} f_i$ and (A-1):

$$R'_1(1) = \sum_{i \in \text{FC}} \frac{f_i}{\Lambda} \leq 1 = R'_1(0).$$

For $d > 1$, using (A-2),

$$R'_1(d) = \sum_{i \in \text{FC}, i \geq d} \frac{f_i}{\Lambda} \leq \sum_{i \in \text{FC}, i \geq d-1} \frac{f_i}{\Lambda} = R'_1(d - 1).$$

- Induction step: Let $m > 0$ and assume $R'_m(d)$, $d > 0$ is decreasing on d ; it has to be shown that $R'_{m+1}(d) \leq R'_{m+1}(d - 1)$, $d > 0$.

For $d = 1$, using (A-3), $\Lambda \geq \sum_{i \in \text{FC}} f_i$, $R'_m(1) \leq 1$, and (A-1),

$$R'_{m+1}(1) = \left(1 - \frac{1}{\Lambda} \cdot \sum_{i \in \text{FC}} f_i\right) \cdot R'_m(1) + \sum_{i \in \text{FC}} \frac{f_i}{\Lambda} \cdot R'_m(0) \leq 1 - \frac{1}{\Lambda} \cdot \sum_{i \in \text{FC}} f_i + \sum_{i \in \text{FC}} \frac{f_i}{\Lambda} = 1 = R'_{m+1}(0).$$

For $d > 1$, using (A-3), $\Lambda \geq \sum_{i \in \text{FC}} f_i$, and the induction hypothesis,

$$\begin{aligned} R'_{m+1}(d) &= \left(1 - \frac{1}{\Lambda} \cdot \sum_{i \in \text{FC}} f_i\right) \cdot R'_m(d) + \sum_{i \in \text{FC}} \frac{f_i}{\Lambda} \cdot R'_m(\max\{0, d - i\}) \\ &\leq \left(1 - \frac{1}{\Lambda} \cdot \sum_{i \in \text{FC}} f_i\right) \cdot R'_m(d - 1) + \sum_{i \in \text{FC}} \frac{f_i}{\Lambda} \cdot R'_m(\max\{0, d - i - 1\}) \\ &= R'_{m+1}(d - 1). \end{aligned}$$

Q.E.D.

B. Proof of Proposition 1

Let $\lambda_{a,f}^i$ be the contribution to $\lambda_{a,f}$ associated with failure bags $e \in E_i$. Then, $\lambda_{a,f}^i \leq f_i$. If $i < d$, $\lambda_{a,f}$ does not have any contribution $\lambda_{a,f}^i$ because $a \in U_d$ and a failure bag $e \in E_i$ reduces the failure distance by at most i . Therefore,

$$R_1(a) = \sum_{i \in \text{FC}, i \geq d} \frac{\lambda_{a,f}^i}{\Lambda}. \quad (\text{A-4})$$

Let k be the number of failed components of the system in state a . Since f is absorbing, for $m > 1$,

$$\begin{aligned} R_m(a) &= \left(1 - \frac{\lambda_a}{\Lambda}\right) \cdot R_{m-1}(a) \\ &+ \sum_{i \in \text{FC}, i \geq d} \left[\frac{\lambda_{a,f}^i}{\Lambda} + \sum_{d'=1}^d \sum_{b \in U_{k+i}, d'} \frac{\lambda_{a,b}}{\Lambda} \cdot R_{m-1}(b) \right] \\ &+ \sum_{i \in \text{FC}, i < d} \sum_{d'=d-i}^d \sum_{b \in U_{k+i}, d'} \frac{\lambda_{a,b}}{\Lambda} \cdot R_{m-1}(b). \end{aligned} \quad (\text{A-5})$$

The proof is by induction on m .

- Base case ($m = 1$): We show that $R_1(a) \leq R'_1(\min\{d, \tilde{L}\})$, with $a \in U_d$, $d > 0$. Using (A-4), $\lambda_{a,f}^i \leq f_i$, and (A-2),

$$\begin{aligned} R_1(a) &= \sum_{i \in \text{FC}, i \geq d} \frac{\lambda_{a,f}^i}{\Lambda} \leq \sum_{i \in \text{FC}, i \geq d} \frac{f_i}{\Lambda} \\ &\leq \sum_{i \in \text{FC}, i \geq \min\{d, \tilde{L}\}} \frac{f_i}{\Lambda} = R'_1(\min\{d, \tilde{L}\}). \end{aligned}$$

- Induction step: Let $m > 0$ and assume

$R_m(a) \leq R'_m(\min\{d, \tilde{L}\})$, $a \in U_d$, $d > 0$; it has to be shown that $R_{m+1}(a) \leq R'_{m+1}(\min\{d, \tilde{L}\})$, $a \in U_d$, $d > 0$. Using (A-5), $\Lambda \geq \lambda_a$ (which implies $1 - \lambda_a/\Lambda \geq 0$), and the induction hypothesis,

$$\begin{aligned} R_{m+1}(a) &\leq \left(1 - \frac{\lambda_a}{\Lambda}\right) \cdot R'_m(\min\{d, \tilde{L}\}) + \sum_{i \in \text{FC}, i \geq d} \\ &\cdot \left[\frac{\lambda_{a,f}^i}{\Lambda} + \sum_{d'=1}^d \sum_{b \in U_{k+i}, d'} \frac{\lambda_{a,b}}{\Lambda} R'_m(\min\{d', \tilde{L}\}) \right] \\ &+ \sum_{i \in \text{FC}, i < d} \sum_{d'=d-i}^d \sum_{b \in U_{k+i}, d'} \frac{\lambda_{a,b}}{\Lambda} R'_m(\min\{d', \tilde{L}\}). \end{aligned}$$

Using $R'_m(\min\{d', \tilde{L}\}) \leq 1$, lemma 1, and that for $i \in \text{FC}$ and $d > 1$

$$\sum_{d'=d-j}^d \sum_{b \in U_{k+i}, d'} \lambda_{a,b} \leq \lambda_{a, U^{k+i}}, \quad 0 \leq j < d,$$

we have

$$\begin{aligned} R_{m+1}(a) &\leq \left(1 - \frac{\lambda_a}{\Lambda}\right) \cdot R'_m(\min\{d, \tilde{L}\}) \\ &+ \sum_{i \in \text{FC}, i \geq d} \left[\frac{\lambda_{a,f}^i}{\Lambda} + \sum_{d'=1}^d \sum_{b \in U_{k+i}, d'} \frac{\lambda_{a,b}}{\Lambda} \right] \\ &+ \sum_{i \in \text{FC}, i < d} \left[R'_m(\min\{d-i, \tilde{L}\}) \cdot \sum_{d'=d-i}^d \sum_{b \in U_{k+i}, d'} \frac{\lambda_{a,b}}{\Lambda} \right] \\ &\leq \left(1 - \frac{\lambda_a}{\Lambda}\right) \cdot R'_m(\min\{d, \tilde{L}\}) + \sum_{i \in \text{FC}, i \geq d} \frac{\lambda_{a,f}^i + \lambda_{a, U^{k+i}}}{\Lambda} \\ &+ \sum_{i \in \text{FC}, i < d} R'_m(\min\{d-i, \tilde{L}\}) \cdot \frac{\lambda_{a, U^{k+i}}}{\Lambda}. \end{aligned} \quad (\text{A-6})$$

Since $\lambda_a = \lambda_{a,f} + \sum_{i \in \text{FC}} \lambda_{a, U^{k+i}}$,

$$\frac{\lambda_a}{\Lambda} = \sum_{i \in \text{FC}, i \geq d} \frac{\lambda_{a,f}^i + \lambda_{a, U^{k+i}}}{\Lambda} + \sum_{i \in \text{FC}, i < d} \frac{\lambda_{a, U^{k+i}}}{\Lambda}. \quad (\text{A-7})$$

Combining (A-6) and (A-7),

$$\begin{aligned} R_{m+1}(a) &\leq R'_m(\min\{d, \tilde{L}\}) \\ &+ \sum_{i \in \text{FC}, i \geq d} [1 - R'_m(\min\{d, \tilde{L}\})] \cdot \frac{\lambda_{a,f}^i + \lambda_{a, U^{k+i}}}{\Lambda} \\ &+ \sum_{i \in \text{FC}, i < d} [R'_m(\min\{d-i, \tilde{L}\}) - R'_m(\min\{d, \tilde{L}\})] \\ &\cdot \frac{\lambda_{a, U^{k+i}}}{\Lambda}. \end{aligned}$$

For $i \geq d$, $\lambda_{a,f}^i + \lambda_{a, U^{k+i}} \leq f_i$; for $i < d$, $\lambda_{a, U^{k+i}} < f_i$; then, using $R'_m(\min\{d, \tilde{L}\}) \leq 1$, lemma 1 (which guarantees $R'_m(\min\{d-i, \tilde{L}\}) - R'_m(\min\{d, \tilde{L}\}) \geq 0$, $i > 0$), (A-1), and (A-3),

$$\begin{aligned} R_{m+1}(a) &\leq R'_m(\min\{d, \tilde{L}\}) \\ &+ \sum_{i \in \text{FC}, i \geq d} [1 - R'_m(\min\{d, \tilde{L}\})] \cdot \frac{f_i}{\Lambda} \\ &+ \sum_{i \in \text{FC}, i < d} [R'_m(\min\{d-i, \tilde{L}\}) \\ &\quad - R'_m(\min\{d, \tilde{L}\})] \cdot \frac{f_i}{\Lambda} \\ &= \left(1 - \frac{1}{\Lambda} \cdot \sum_{i \in \text{FC}} f_i\right) \cdot R'_m(\min\{d, \tilde{L}\}) \\ &+ \sum_{i \in \text{FC}} \frac{f_i}{\Lambda} \cdot R'_m(\max\{0, \min\{d-i, \tilde{L}\}\}) \\ &\leq \left(1 - \frac{1}{\Lambda} \cdot \sum_{i \in \text{FC}} f_i\right) \cdot R'_m(\min\{d, \tilde{L}\}) \end{aligned}$$

$$\begin{aligned}
 & + \sum_{i \in \text{FC}} \frac{f_i}{\Lambda} \cdot R'_m(\max\{0, \min\{d, \tilde{L}\} - i\}) \\
 & = R'_{m+1}(\min\{d, \tilde{L}\}).
 \end{aligned}$$

Q.E.D.

C. Proof of Proposition 2

The following notation is used in the proof:

$$\psi(m, x) \equiv Y_{m-1} \in G \wedge Y_m = x,$$

$$\psi'(m, x) \equiv Y'_{m-1} \in G \wedge Y'_m = x.$$

Y can enter f through U or directly from G . Because f is absorbing, conditioning the entry of Y in f through U to the step in which Y enters U and the entry state, and using (5):

$$\begin{aligned}
 & \Pr\{Y_n = f\} \\
 & = \sum_{m=1}^{n-1} \sum_{a \in U} \Pr\{\psi(m, a)\} \cdot \Pr\{Y_n = f | Y_m = a\} \\
 & \quad + \sum_{m=1}^n \Pr\{\psi(m, f)\} \\
 & = \sum_{m=1}^{n-1} \sum_{a \in U} \Pr\{\psi(m, a)\} \cdot R_{n-m}(a) \\
 & \quad + \sum_{m=1}^n \Pr\{\psi(m, f)\}.
 \end{aligned}$$

Since, for $a \in U$, $1 \leq \tilde{d}(a) \leq d(a)$ and $\tilde{d}(a) \leq \tilde{L}$, U_d can be partitioned as:

$$U_d = \bigcup_{i=1}^{\min\{d, \tilde{L}\}} \tilde{U}_{d,i}.$$

Then, since $\tilde{L} = \tilde{d}(o) \leq d(o) = L$,

$$U = \bigcup_{d=1}^L U_d = \bigcup_{d=1}^L \bigcup_{i=1}^{\min\{d, \tilde{L}\}} \tilde{U}_{d,i} = \bigcup_{i=1}^{\tilde{L}} \bigcup_{d=i}^L \tilde{U}_{d,i}.$$

Using this partition of U and proposition 1,

$$\begin{aligned}
 & \Pr\{Y_n = f\} \\
 & = \sum_{m=1}^{n-1} \sum_{i=1}^{\tilde{L}} \sum_{d=i}^L \sum_{a \in \tilde{U}_{d,i}} \Pr\{\psi(m, a)\} \cdot R_{n-m}(a) \\
 & \quad + \sum_{m=1}^n \Pr\{\psi(m, f)\} \\
 & \leq \sum_{m=1}^{n-1} \sum_{i=1}^{\tilde{L}} \sum_{d=i}^L \sum_{a \in \tilde{U}_{d,i}} \Pr\{\psi(m, a)\} \cdot R'_{n-m}(\min\{d, \tilde{L}\}) \\
 & \quad + \sum_{m=1}^n \Pr\{\psi(m, f)\}.
 \end{aligned}$$

Using lemma 1, $\tilde{U}_i = \bigcup_{d=i}^L \tilde{U}_{d,i}$, the relations between Y and Y' , and (4):

$$\begin{aligned}
 & \Pr\{Y_n = f\} \\
 & \leq \sum_{m=1}^{n-1} \sum_{i=1}^{\tilde{L}} \sum_{d=i}^L \sum_{a \in \tilde{U}_{d,i}} \Pr\{\psi(m, a)\} \cdot R'_{n-m}(i) \\
 & \quad + \sum_{m=1}^n \Pr\{\psi(m, f)\} \\
 & = \sum_{m=1}^{n-1} \sum_{i=1}^{\tilde{L}} \sum_{a \in \tilde{U}_i} \Pr\{\psi(m, a)\} \cdot R'_{n-m}(i) \\
 & \quad + \sum_{m=1}^n \Pr\{\psi(m, f)\} \\
 & = \sum_{m=1}^{n-1} \sum_{i=1}^{\tilde{L}} \Pr\{\psi'(m, u_i)\} \cdot R'_{n-m}(i) \\
 & \quad + \sum_{m=1}^n \Pr\{\psi'(m, f)\} \\
 & = \sum_{m=1}^{n-1} \sum_{i=1}^{\tilde{L}} \Pr\{\psi'(m, u_i)\} \cdot \Pr\{Y'_n = u_0 | Y'_m = u_i\} \\
 & \quad + \sum_{m=1}^n \Pr\{\psi'(m, f)\} \\
 & = \Pr\{Y'_n = u_0\} + \Pr\{Y'_n = f\} = \Pr\{Y'_n \in \{u_0, f\}\}.
 \end{aligned}$$

Q.E.D.

D. Proof of Theorem 1

Using (2) and $\Pr\{Y_0 = f\} = \Pr\{X(0) = f\} = 0$,

$$\text{ur}(t) = \Pr\{X(t) = f\} = \sum_{n=1}^{\infty} \text{poim}(n; \Lambda \cdot t) \cdot \Pr\{Y_n = f\}.$$

Using proposition 2,

$$\Pr\{Y'_0 \in \{u_0, f\}\} = \Pr\{X'(0) \in \{u_0, f\}\} = 0,$$

(3), and (1),

$$\begin{aligned}
 \text{ur}(t) & \leq \sum_{n=1}^{\infty} \text{poim}(n; \Lambda \cdot t) \cdot \Pr\{Y'_n \in \{u_0, f\}\} \\
 & = \sum_{n=0}^{\infty} \text{poim}(n; \Lambda \cdot t) \cdot \Pr\{Y'_n \in \{u_0, f\}\} \\
 & = \Pr\{X'(t) \in \{u_0, f\}\} = [\text{ur}(t)]_{\text{ub}}.
 \end{aligned}$$

Q.E.D.

REFERENCES

- [1] V. Suñé and J. A. Carrasco, "A method for the computation of reliability bounds for nonrepairable fault-tolerant systems," in *Proc. 5th IEEE Int'l Symp. Modeling, Analysis, and Simulation of Computers and Telecommunication Systems (MASCOTS'97)*, Haifa, Israel, 1997, pp. 221–228.
- [2] M. A. Boyd, M. Veeraraghavan, J. B. Dugan, and K. Trivedi, "An approach to solving large reliability models," in *Proc. AIAA/IEEE 8th Conf. Embedded Digital Avionics*, 1988, pp. 243–250.

- [3] S. M. Ross, *Stochastic Processes*: John Wiley & Sons, 1983.
- [4] J. L. Peterson, "Computation sequence sets," *J. Computer and System Sciences*, vol. 13, pp. 223–252, Aug. 1976.
- [5] R. A. Sahner and K. S. Trivedi, "Reliability modeling using SHARPE," *IEEE Trans. Reliability*, vol. R-36, pp. 186–193, Jun. 1987.
- [6] D. Lee, J. Abraham, D. Rennels, and G. Gilley, "A numerical technique for the hierarchical evaluation of large, closed fault-tolerant systems," in *Proc. 3rd Working Conf. Dependable Computing for Critical Applications*, 1992, pp. 95–114.
- [7] J. B. Dugan, "Fault trees and imperfect coverage," *IEEE Trans. Reliability*, vol. 38, pp. 177–185, Jun. 1989.
- [8] S. A. Doyle and J. B. Dugan, "Dependability assessment using binary decision diagrams (BDD's)," in *Proc. 25th IEEE Int'l Symp. Fault-Tolerant Computing*, 1995, FTCS-25, pp. 249–258.
- [9] J. Carrasco and V. Suñé, "An algorithm to find minimal cuts of coherent fault trees with event classes, using a decision tree," *IEEE Trans. Reliability*, vol. 48, pp. 31–41, Mar. 1999.
- [10] A. Rosenthal, "A computer scientist looks at reliability computations," in *Reliability and Fault Tree Analysis*, A. Barlow, A. Fussell, and A. Singpurwalla, Eds: SIAM, 1975, pp. 133–152.
- [11] D. P. Heyman and H. J. Sobel, *Stochastic Models in Operations Research*: McGraw-Hill, 1982, vol. I.
- [12] T. Kohda, E. J. Henley, and K. Inoue, "Finding modules in fault trees," *IEEE Trans. Reliability*, vol. 38, pp. 165–176, Jun. 1989.
- [13] Y. Dutuit and A. Rauzy, "A linear-time algorithm to find modules of fault trees," *IEEE Trans. Reliability*, vol. 45, pp. 422–425, Sep. 1996.
- [14] V. Suñé and J. Carrasco. (1999, Jan.) A failure-distance based method to bound the reliability of nonrepairable fault-tolerant systems without the knowledge of minimal cuts. Universitat Politècnica de Catalunya. [Online] Available by ftp from . Available: <ftp://ftp-eel.upc.es/techreports>
- [15] D. Gross and D. Miller, "The randomization technique as a modeling tool and solution procedure for transient Markov processes," *Operations Research*, vol. 32, pp. 343–361, Mar.–Apr. 1984.

Víctor Suñé is currently Assistant Professor at the "Departament d'Enginyeria Electrònica" of UPC ("Universitat Politècnica de Catalunya"). His research focuses on bounding techniques for dependability estimation of fault-tolerant systems, which was his dissertation topic. He received the Engineer degree (1995) in industrial engineering and the Doctor Engineer degree (2000) in industrial engineering, both from UPC.

Juan A. Carrasco has been Associate Professor at the "Departament d'Enginyeria Electrònica" of UPC ("Universitat Politècnica de Catalunya") since 1988. His research focuses on modeling of digital systems and, more specifically, fault-tolerant systems. He received the Engineer degree (1981) in industrial engineering from UPC; M.S. (1987) in computer science from Stanford University; and the Doctor Engineer degree (1987) from UPC. He co-leads a research group on Reliability and Fault-Tolerance of Electronic Systems at the Dep't d'Enginyeria Electrònica of UPC. He has been member of the program committee of several international conferences.